VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Bakalářská práce

2015 **Robin Prašivka**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Konfigurační rozhraní pro aplikace v mikropočítači

Configuration Interface for Microcontroler's Applications

2015 **Robin Prašivka**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Robin Prašivka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Konfigurační rozhraní pro aplikace v mikropočítači
Configuration Interface for Microcontroler's Applications

Zásady pro vypracování:

Navrhněte jednoduché textové uživatelské rozhraní pro mikropočítače, připojené přes sériové rozhraní k PC tak, aby bylo ovládání možno provádět přes program hyperterminal nebo minicom a využije maximum možností, které tyto emulátory terminálů mají. Navržené rozhraní nesmí ovlivňovat běh stávající aplikace.

1. Najděte a porovnejte dostupná textová rozhraní, vhodná pro mikropočítače.
2. Seznamte se s vlastnostmi textových terminálů a ověřte funkcionalitu dostupných emulátorů terminálů.
3. Navrhněte koncepci uživatelského rozhraní pro mikropočítač.
4. Implementujte uživatelské rozhraní jako knihovnu.
5. Realizujte prototypovou aplikaci pomocí navrženého rozhraní.
6. Zhodnoťte paměťovou a procesorovou náročnost navrženého rozhraní, jeho stabilitu a složitost integrace do dalších aplikací.

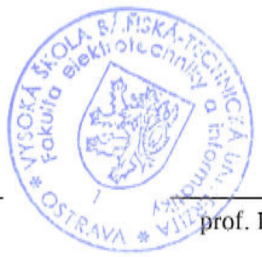
Seznam doporučené odborné literatury:

[1] Stanislav Pechal, Monolitické mikropočítače, BEN - Technická literatura, ISBN:80-86056-30-9
[2] Text terminal HOWTO, http://www.linuxdoc.org/HOWTO/Text-Terminal-HOWTO.html

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Olivka, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 07. května 2015

.....

Rád bych poděkoval svému vedoucímu práce, panu Petru Olivkovi, za poskytnutá doporučení, čas strávený na konzultacích a pozitivní přístup.

Dále bych chtěl poděkovat rodině a přátelům za trpělivost, pochopení a nezbytnou podporu při studiu.

Děkuji.

Abstrakt

Tato práce je zaměřená na textová komunikační rozhraní použitelná pro mikropočítače. Porovnává jejich možnosti i nároky kladené jak na hardware, tak na znalosti uživatele. Dále zkoumá možnosti dostupných emulátorů textových terminálů a jejich použitelnost ve spolupráci s textovým rozhraním mikropočítače. Druhá část popisuje vývoj jednoduchého textového rozhraní a jeho realizaci. Závěr je věnován ověření funkčnosti vytvořeného rozhraní, jeho stability, paměťovým nárokům a složitosti implementace do dalších aplikací.

Klíčová slova

mikropočítač, ascii tabulka, ANSI/VT100, escape sekvence, terminál, textové uživatelské rozraní

Abstract

This work is focused on text communication interface usable for microcomputers. Compares their possibilities and requirements on the hardware and the user's knowledge. It also explores available text terminal emulators and their applicability in the text interface. The second part describes the development of a simple text interface and its implementation. Finally it's verify the functionality of the interface, its stability, memory requirements and complexity of implementation into other applications.

Keywords

microcomputer, ascii table, ANSI/VT100, escape sequences, terminal, text user interface

Seznam zkratek

ASCII	- American Standard Code for Information Interchange
ANSI	- American National Standards Institute
CLI	- Command Line Interface
CSI	- Control Sequence Introducer
DEC	- Digital Equipment Corporation
SRAM	- Static Random Access Memory
SSH	- Secure Shell
TCP/IP	- Transmission Control Protocol/Internet Protocol
TUI	- Text User Interface
UART	- Universal Asynchronous Receiver/Transmitter
UTF-8	- Universal Transformation Format

Obsah

1. Úvod

Mikropočítače dnes nacházejí stále větší uplatnění. Dostatečný výkon a příznivá cena umožňují jejich nasazení v širokém spektru aplikací. Od jednoduchých regulací až po řízení složitých procesů. Jejich interakce s uživatelem se odvíjí od aplikace, ve které jsou nasazeny.

U aplikací, jako jsou různé generátory, převodníky nebo zcela autonomní systémy, nemusí být použité žádné prostředky pro komunikaci s uživatelem. Mikropočítač pouze vyhodnotí vstupní parametry a podle nich dále reaguje. Jeho stav nebo nefunkčnost se vyhodnocuje až v nadřazeném systému.

Nejsnadnější možnost, jak zobrazit nějaký stav aplikace, či přijmout povel od uživatele, mohou být různé stavové kontrolky a tlačítka napojené na porty mikropočítače. Počet takových portů je často omezený, proto se používají jen k zobrazení základních stavů, jako zapnuto, vypnuto a podobně.

Dalším rozšířením může být například připojení různých typů displejů a klávesnic, které možnosti komunikace rozšiřují. Lze říci, že ke komunikaci mikropočítače s uživatelem lze využít všechna rozhraní, která daný mikropočítač nabízí. Mezi tato rozhraní řadíme i sériová rozhraní, přes které probíhá komunikace pomocí textových zpráv.

Zařízení, které umožňuje zprostředkovat takovou komunikaci, se nazývá textový terminál. Obvykle se skládá ze zobrazovací jednotky, monitoru a klávesnice. Dříve se terminály používaly pro vzdálené ovládání počítače. Dnes se jejich funkce simuluje pomocí osobního počítače. Jejich funkce spočívá v zobrazení příchozích znaků a odesílání kódů stisknutých kláves. Na tomto principu jsou postavena textová uživatelská rozhraní.

2. Textová rozhraní

Textová rozhraní mohou mít různou podobu. Od základního psaní příkazů až po různé nabídky členěné do oken. Jednotlivé typy se liší svými hardwarovými nároky i komfortem ovládání.

2.1 Příkazový řádek (CLI)

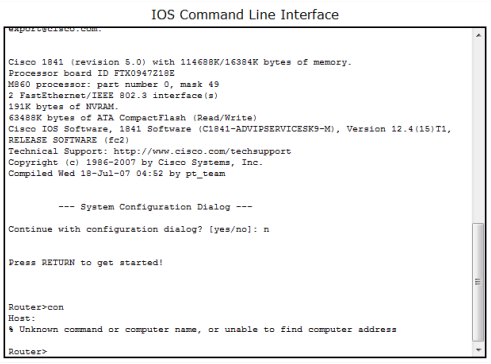
K nejjednodušší formě textového komunikačního rozhraní patří příkazový řádek. Když je systém připraven přijmout příkaz, zobrazuje tzv. výzvu (prompt) a to ve formě blikajícího kurzoru nebo nějaké informace o stavu systému. Uživatel zadá příkaz ve formě sekvence znaků a potvrdí stiskem klávesy Enter. Systém příkaz dekóduje a provede. Výsledky vykonání příkazu se zobrazí v okně terminálu. Uživatel může být také vyzván k zadání dalších parametrů. Ukončení provádění příkazu je signalizováno zobrazením výzvy k dalšímu příkazu.

Výhody použití příkazového řádku:

- Nízké hardwarové nároky a přenosové rychlosti. Příkazový řádek má nejnižší HW nároky ze všech textových rozhraní. Systém nemusí uchovávat v paměti žádné nastavení rozhraní a vykreslovat možnosti voleb, pouze reaguje na zadané příkazy.
- Rychlost práce. Zkušený uživatel přímo zadává příkazy, které potřebuje, odpadá nutnost procházet různé nabídky. Zároveň se minimalizuje možnost nechtěného spuštění jiné funkce, každý příkaz je jasně definovaný.
- Možnosti kombinace příkazu. Uživatel má možnost zvolit více kombinací příkazů než jaké nabízí např. menu. Některé aplikace umožňují uložení takové sekvence a její provedení pomocí jejího vyvolání.

Nevýhoda použití příkazového řádku je znalost příkazů. Uživatel se musí v začátcích naučit příkazy, které chce využívat. To zabere mnohem více času, než při procházení v menu. Uživatel by měl vědět, co chce dělat a jak to chce udělat. Rozhraní mu nenabízí žádné možnosti, ze kterých by si mohl vybrat.

Příkazový řádek je pro svou univerzálnost poměrně rozšířený. Mezi nejznámější patří příkazový řádek v operačních systémech počítačů či IOS u síťových prvků CISCO.



Obr. 1: Příkazový řádek IOS

2.2 Nabídkové menu

U tohoto typu komunikace se na obrazovku terminálu vypisuje nabídka možnosti ve formě menu. Uživatel si vybírá požadovaný příkaz.

Nejjednodušší formou je zobrazení nabídky v řádcích. Řádky jsou obvykle číslovány nebo označeny malými písmeny. Uživatel je vyzván k zadání označení vybraného řádku. Po zadání příslušné volby se příkaz provede a je opět zobrazena nabídka dostupných příkazů. Stejnou formou je řešeno i zanořování do nabídek, kdy se zobrazuji pouze aktuálně dostupné položky. Tato forma menu je nejméně náročná na rychlost komunikace se systémem.

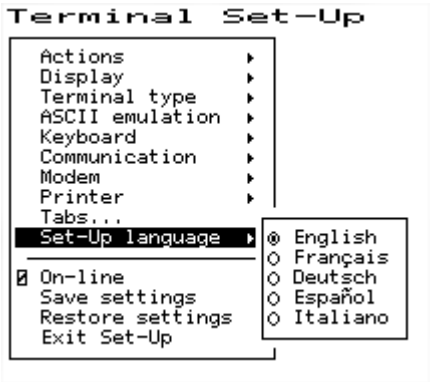
Komfortnější forma menu zobrazuje nabídku příkazů. Mikropočítač reaguje na stisknutí akčních kláves a dynamický mění vzhled menu. Toto řešení je již více náročné na komunikaci mezi aplikací a terminálem. Na obrazovku se neposílá jen nabídka příkazů, ale při každém pohybu v menu dochází k překreslování dotčených řádků.

Výhody použití menu:

- Nízké hardwarové nároky. Aplikace má v paměti uloženou pouze strukturu menu s odkazy na jednotlivé funkce. Ke komunikaci dochází hlavně při vykreslování nabídky.
- Komfort obsluhy. Uživatel si vybírá z nabízených možností. Nemusí znát žádné příkazy, pouze si intuitivně vybírá z nabídky. Ta bývá logicky seřazena v jednotlivých úrovních menu.

Při velkém množství nabízených funkcí se menu stává nepřehledné. Při hlubším zanoření je časově náročné provést více různých příkazů za sebou.

Nabídkové menu nachází uplatnění v menších aplikacích, jako jsou různé regulátory nebo převodníky. Vhodné je tam, kde je potřeba občas nastavit nebo sledovat nějaké provozní parametry. Předem připravenou nabídkou funkcí umožňuje efektivně řídit běžící aplikaci.



Obr. 2: Nabídkové menu

2.3 Pokročilá textová rozhraní

Nabídka funkci bývá rozdělena do jednotlivých oken, mnohdy barevně rozlišených. Nabízí celou řadu pokročilých objektů, jako jsou menu, tlačítka, okna, seznamy atd.

Terminálové okno pracuje v textovém režimu. Je rozděleno do pevného rastru řádek a sloupců. V každé pozici může být zobrazený pouze jeden znak z dané množiny. Pomocí těchto znaků jsou dále vytvářené grafické prvky na obrazovce.

V rámci textových rozhraní nabízí nejvyšší komfort pro uživatele. Dovede přehledně pokrýt širokou škálu funkcí, aniž by kladlo vysoké nároky znalosti uživatele. U jednotlivých položek může být zobrazená nápověda s popisem funkce a integritních omezení.

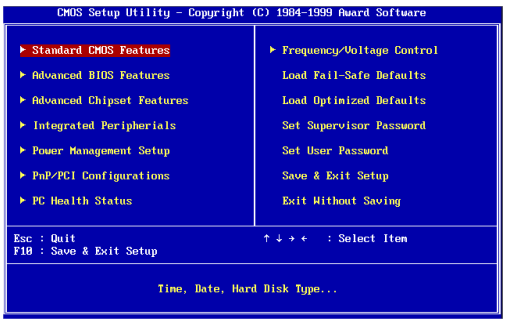
Tyto vlastnosti jsou vykoupeny vyššími nároky na procesor i paměť mikropočítače. Rovněž klade větší požadavky na rychlost komunikace s terminálem, kde při procházení nabídek se překreslují celé oblasti obrazovky. Plně využívá možnosti terminálu pro nastavování barev i pohybu kurzoru.

Výhody použití:

- Komfort ovládání. Přehledné zpracování, zobrazování nápovědy a pokročilé ovládací prvky pomáhají uživateli orientovat se v nabídce. Uživatel si vybírá z připravených nabídek bez potřeby speciálních znalosti příkazů.
- Rozsah. Využitím kombinace menu a oken dokáže přehledně pokrýt širokou škálu implementovaných funkcí a nastavení.

Nevýhodou mohou být vyšší hardwarové nároky. Svým grafickým ztvárněním klade vysoké nároky na paměť, početní výkon mikropočítače a objem přenášených dat s terminálem.

Pokročilé textové uživatelské rozhraní můžeme nalézt u složitějších aplikací a v různých vestavěných systémech. Příkladem by mohl být teletext televize či BIOS u základních desek počítačů.



Obr. 3: Pokročilá textová rozhraní

3. Textový terminál

Terminál slouží pro vzdálenou komunikaci s počítačem. Skládá se ze zobrazovací jednotky a klávesnice. Ve spojení s mikropočítačem může sloužit k jeho ovládání, nastavování parametrů a sledování procesních hodnot v běžícím programu.

Při aplikacích, které běží samostatně a zásah obsluhy potřebují jen při spouštění, poruše nebo změně nastavení, lze využít terminál jako vhodný prostředek pro komunikaci s uživatelem. Mikropočítače jsou obvykle vybaveny některým sériovým rozhraním, které může být použito pro připojení k terminálu. Terminál pak není součásti aplikace, čímž se snižují rozměry i celkové náklady na zařízení.

3.1 Komunikace s textovým terminálem

Textový terminál má obrazovku rozdělenou do pevného rastru znaků. Grafická podoba znaků je uložena přímo v terminálu. Každému znaku je přiřazen ASCII kód. Komunikace s textovým terminálem probíhá tak, že mikropočítač posílá sekvence kódu jednotlivých znaků, které mají být zobrazeny. Terminal zpátky posílá kódy stisknutých kláves.

Standardem se stala ASCII tabulka znaků, viz Obr. 4. Obsahuje tisknutelné znaky a řídící kódy, které byly původně určené k řízení externích zařízení (např. tiskárny). Původní ASCII kód je sedmibitový a obsahuje tedy 127 znaků. Kvůli potřebě více znaků se následně rozšířil na osm bitů. Znaky 127 až 255 se mohou lišit podle jazyka a prostředí, kde je tabulka používána.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	⌂

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	|	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	ø	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	Ø	189	BD	╜	221	DD	▌	253	FD	²
158	9E	×	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	⌂

Obr. 4: ASCII tabulka znaků [1]

3.2 Ovládání terminálu

Prostředí terminálu je ovládáno připojeným počítačem. Ten neposílá terminálů pouze znaky, které mají být zobrazeny, ale i příkazy jak se má chovat. Existují dvě možnosti, jak to může udělat. Jedná se o řídící znaky a tzv. řídící (escape) sekvence [2].

3.2.1 Řídící znaky

Řídící znaky se skládají z prvních 32 znaků ASCII tabulky [2]. Slouží k řízení přenosů dat, nastavení formátu výstupu a dalším účelům. Tyto znaky byly původně určeny k řízení tiskárny nebo jiných výstupních zařízení a dnes se z nich prakticky využívá jen malá část. Mezi nejznámější patří LF – Line Feed (odřádkování), HT – Horizontal tab (tabulátor) nebo CR – Carriage Return (návrat vozíku). Použití těchto znaků je obvykle dané, ale v jednotlivých systémech se může lišit.

3.2.2 Řídící (escape) sekvence

Sekvence kódu, které jsou určené pro kontrolu formátování, barvy a dalších vlastností na výstupu textových terminálů. Tyto sekvence jsou zasílány terminálu spolu s daty. Terminál je rozpozná a přeloží si je jako příkazy pro nastavení. U jednotlivých typů textových terminálů se mohou lišit. Organizace ANSI definovala standart těchto sekvencí a většina používaných textových terminálů umí pracovat alespoň s částí z nich.

Řídících sekvencí je celá řada. Pro formátování výstupu textového terminálu se využívají tzv. CSI kódy, viz Tabulka 1. Takový příkaz začíná dvojicí znaků ESC [.

ECS[A	- kurzor nahoru
ESC[B	- kurzor dolů
ESC[C	- kurzor dopředu
ESC[D	- kurzor dozadu
ESC[n;mf	- kurzor do pozice *n* (řádek), *m* (sloupec)
ESC[2J	- smazání celé obrazovky
ESC[2K	- smazání řádku
ESC[x;x;xm	- barvy a atributy textu Černá (0), Červená (1), Zelená (2), Žlutá (3), Modrá (4), Fialová (5), Azurová (6), Bílá (7) a Defaultní (9) K barvě popředí se připočte 30 a k barvě pozadí 40. Světlý (1), Tmavý (2), Podtržený (4), Blikající (5), Inverzní (7) a Skrytý (8)
ESC[0m	- reset nastavení barev a atributů textu

Tabulka 1: Vybrané řídící sekvence [3]

3.3 Emulátory textových terminálů

Emulátor je program, který napodobuje systém terminálu pomocí osobního počítače. Ke komunikaci se používá buď přímé spojení kabelem přes sériový port, nebo přes modem. V jiných případech může k výměně dat docházet pomocí paketu přes rozhraní síťové karty. Pro spojení s mikropočítači se nejčastěji využívá právě sériový port.

Nejčastěji podporovaným standardem u textových terminálů je VT100/ANSI. VT100 je video terminál vyvinutý společnosti DEC. Dnes se toto označení používá pro kolekci řídících sekvencí, které daný emulátor podporuje.

Z volně dostupných emulátorů textových terminálu jsem vybral:

- Hyperterminál
  Program společnosti Microsoft. Až do Windows XP byl standartní součásti operačního systému. Program se používá pro komunikaci s dalším počítačem nebo zařízením přes sériový port, telefonní modem nebo síťovou kartu pomocí TCP/IP [4]. Lze použít ke konfiguraci zařízení, vytváření telefonních spojení nebo testování modemů. Umožňuje přenos souborů, zaznamenávat a ukládat probíhající komunikaci. Jako výstup může být použita obrazovka nebo tiskárna. Podporuje emulaci terminálu VT52, VT100 , ANSI, TTY a další.

- Minicom
  Minicom je textový komunikační program pro sériový port. Používá se ke komunikaci s externími zařízeními přes RS-232 jako mobily, směrovače a konzole sériového portu [5]. Pracuje v textovém okně konzole systému. Umožňuje posílat soubory, ukládat záznamy a práci se skripty. Parametry lze nastavit přes menu nebo pomocí přepínačů při spuštění programu. Podporuje emulaci VT102. Dostupný pro Linux.

- PuTTY
  Primárně TelNet a SSH-2 klient bez nutnosti instalace. Podporuje i komunikaci přes RS232. Dokáže emulovat terminály VT100, VT400, Linux, XTerm. Pro komunikaci lze zvolit používanou tabulku znaků. Standardně je nastaveno UTF-8. Dostupný pro systémy Linux i Windows.

- TeraTerm
  Emulátor terminálu pro Windows. Dokáže komunikovat přes TCP/IP nebo sériové rozhraní. Používá se jako SSH klient. Emuluje terminály řady VT100.

4. Návrh jednoduchého textového rozhraní pro mikropočítač

Textové rozhraní je jedna z možností pro komunikaci mikropočítače s uživatelem. Připojení k textovému terminálu vyžaduje na straně mikropočítače implementaci takové komunikace, na straně druhé umožní pak připojit takový mikropočítač k libovolnému počítači s emulátorem příslušného terminálu.

4.1 Požadavky na funkcionalitu

Pro svou realizaci textového rozhraní jsem se rozhodl implementovat nabídkové menu. Uživatel bude pomocí šipek procházet jednotlivé položky. Jednotlivé položky budou reprezentovat proměnné, které se používají v běžící aplikaci. Ty bude moci sledovat a případně jim nastavovat nové hodnoty. Další možnosti bude spustit nějakou funkci. Menu bude kaskádové. Bude možno se zanořovat do podnabídek a vracet se zpět.

Navržené textové rozhraní by mělo být snadno implementovatelné do aplikací, bez nutnosti jejích modifikace. Realizované souborem funkcí, pomocí kterých se vytvoří struktura menu a k jednotlivým položkám se přiřadí jednotlivé proměnné nebo funkce. Následně další procedury zajistí funkčnost takto vytvořeného rozhraní.

Rozhraní bude aplikovatelné u více typů mikropočítačů. Použitelné pro co největší množství platforem různých výrobců. Implementované rozhraní by nemělo ovlivňovat stávající aplikaci a pracovat s co nejmenšími nároky na početní výkon a paměť mikropočítače.

4.2 Koncepce uživatelského rozhraní

Pro aplikaci uživatelského rozhraní se nejprve nadefinuje složení menu. Každá obrazovka bude obsahovat nadpis a jednotlivé položky menu. Každá z položek bude odkazovat na proměnnou v programu, funkci nebo bude sloužit jako vstup do dalšího pod-menu. Vytvořená struktura bude uložená v paměti.

Samotná aplikace menu bude pak pracovat s touto uloženou strukturou. Jejím úkolem bude sledovat vstupy z klávesnice, zobrazovat jednotlivá menu na výstup terminálu a pracovat s proměnnými, na které jednotlivé položky aktivní obrazovky menu odkazují. Tato obslužná metoda bude volána cyklicky v pravidelných intervalech.

4.2.1 Vzhled a funkčnost menu

Textový terminál bude zobrazovat vždy jednu obrazovku menu. Ta bude obsahovat název nabídky a výčet položek seřazených jednotlivě pod sebou. Velikost okna se bude dynamicky měnit podle nejdelší položky. V menu se bude procházet pomocí kurzorových kláves. Výběr jednotlivé položky se provede mezerníkem. Klávesou Esc se provede návrat do předchozí nabídky, nebo přeruší zadávání nové hodnoty proměnné. Zadání nové hodnoty proměnné se potvrdí klávesou Enter. Samotné menu lze aktivovat a deaktivovat podle potřeby uživatele.

4.2.2 Implementace programové obsluhy menu

Chod menu bude zajišťovat funkce, která bude cyklicky volaná. To lze zajistit buď v samotné aplikaci běžící v mikropočítači, nebo pomocí implementace přerušení od některého z časovačů.

Tato funkce zkontroluje výstup z klávesnice a změny stavů zobrazovaných proměnných. Následně zašle obrazovce terminálu příkaz k překreslení dotčených položek, či celé obrazovky menu. Pro informace o podobě menu využívá v paměti uloženou strukturu menu.

4.3 Návrh a vývoj

Pro implementaci byl určen programovací jazyk *C*. Většina dnes používaných mikropočítačů má k dispozici pro jazyk *C* překládač. Kód napsaný v jazyce *C* lze snadno použít u různých typů mikropočítačů.

4.3.1 Ovládání textového menu

Pro ovládání menu se používá klávesnice, viz Tabulka 2.

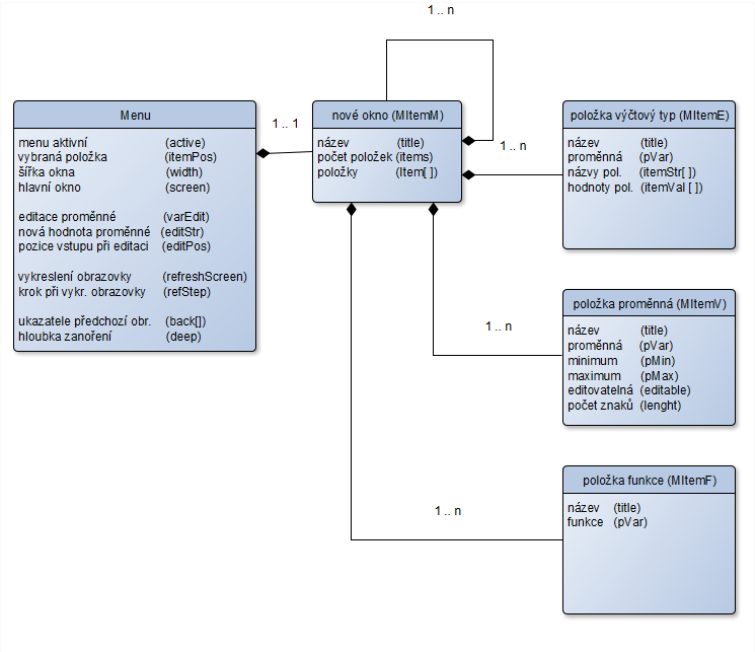
šipky nahoru / dolů	- pohyb mezi položkami v menu
mezerník	- vybrání označené položky v menu (start editace proměnné, spuštění funkce, nebo vyvolání vnořené nabídky)
Ctrl + C	- aktivace a deaktivace menu
Esc	- návrat do předchozí obrazovky, ukončení editace proměnné
Enter	- potvrzení nově zadávané hodnoty proměnné

Tabulka 2: Klávesy pro ovládání menu

Každá položka menu, která pracuje s proměnnou, má definovaný počet znaků, který může interpretace proměnné v menu zabírat. Při editaci má uživatel k dispozici, právě tento počet znaků k zadání nové hodnoty proměnné. Po potvrzení se kontroluje, zda je nová hodnota v definovaném rozmezí a následně nato je fakticky změněná hodnota proměnné. Editaci lze kdykoliv přerušit klávesou Esc.

4.3.2 Implementace menu

Pro zajištění chodu menu v programu je vytvořená struktura *Menu,* která obsahuje parametry s aktuálním stavem textového rozhraní. Vytvořené funkce pro obsluhu menu pak pracují s touto strukturou.



Obr. 5: Struktura menu

Pro počáteční naplnění struktury *Menu* slouží funkce *CreateMenu*, která v parametru obsahuje ukazatel na složení hlavní obrazovky. Tato funkce zjistí šířku okna a nastaví menu jako neaktivní.

```
Menu* CreateMenu(MItemM* screen);
```

Funkčnost celého menu tvoří funkce *RunMenu*, která pracuje s vytvořenou strukturou menu. V programu je třeba zajistit, aby tato funkce byla cyklicky prováděná. Tato funkce kontroluje výstup z klávesnice terminálu a aktualizuje zobrazení menu na obrazovce. Volání této funkce lze provádět v samotném programu aplikace, nebo využít přerušení od některého z časovačů mikropočítače. Frekvence volání této funkce má dopad na rychlost odezvy kláves a překreslování obrazovky terminálu.

```
void RunMenu(Menu*);
```

4.3.3 Definice složení obrazovek menu

Pro vytvoření textového menu je potřeba nadefinovat jeho složení. Menu je složeno z několika oken. Každé z nich má název a několik položek, které obsahuje. Jednou z položek může být i nové okno menu. Položka v menu je v paměti mikropočítače reprezentována strukturou obsahující parametry a ukazatele na funkce, které zajišťují práci s položkou v menu.

Definice nového okna menu

Pro definici nového okna menu slouží funkce *addSubM* s proměnným počtem parametrů. Počet očekávaných položek je určen proměnnou *nrSub*.

```
MItemM* addSubM(char *title, int nrSub, ...);
```

`*title` - Název nového okna menu

`nrSub` - Počet položek v menu

`...` - Výčet jednotlivých položek menu

Funkce vytvoří strukturu *MItemM*, která obsahuje, parametr s počtem položek a pole ukazatelů na jednotlivé položky v novém okně.

Pro každý typ položky menu je společný parametr a několik funkcí, které jsou nutné pro správné zobrazení položky a reakci na aktivaci.

- Název položky
  ```
  char *title;
  ```
- Funkce vracející maximální počet znaků pro zobrazení položky
  ```
  int (*getItemLen)(MItemM* mItemM);
  ```
- Funkce generující řetězec pro zobrazení položky v menu
  ```
  void (*getMenuStr)(MItemM* mItemM,int width,char* str);
  ```
- Funkce, která se volá při aktivaci položky
  ```
  void (*itemActivate)(Menu* menu);
  ```
- Funkce pro aktualizaci zobrazení položky při změně hodnoty proměnné
  ```
  void (*itemRefresh)(Menu* menu, int pos);
  ```

Další proměnné a ukazatele na funkce se mohou lišit podle typu položky, kterou daná struktura reprezentuje.

Definice položky s proměnou

Taková položka v menu zobrazuje aktuální stav proměnné a umožňuje editaci její hodnoty ve stanoveném rozmezí. Připravená struktura *MItemV* je navržená tak, aby se dala použít pro jakýkoli datový typ proměnné. V knihovně menu jsou implementované funkce pro datový typ *integer*.

Pro definici položky, která pracuje s proměnou typu *integer*, se použije funkce:

```
MItemV* addItemI(char *title, int *pi, int lenght, int min, int max,
boolean editable);
```

`*title` - Název nové položky

`*pi` - Ukazatel na proměnnou, se kterou položka pracuje

`lenght` - Počet znaků, které hodnota proměnné v menu zabírá

`min` - minimální hodnota proměnné při editaci

`max` - maximální hodnota proměnné při editaci

`editable` - Je-li hodnota proměnné v menu editovatelná

Struktura *MItemV* obsahuje navíc ukazatele na parametry a funkce, které musí být definované pro konkrétní datový typ proměnné.

- Funkce pro převedení hodnoty proměnné na řetězec znaků pro zobrazení
  ```
  char *(*toString)(MItemV* mItemVar);
  ```

- Funkce, která kontroluje, zda vstupní řetězec lze převést na platnou hodnotu proměnné.
  ```
  boolean(*checkInputStr)(MItemV* mItemVar, char* str, boolean set);
  ```

- Funkce pro porovnání dvou proměnných
  ```
  boolean (*varCompare)(void* var1, void* var2);
  ```

Definice položky spouštějící funkci

Po vyvolání položky je jednou vykonána volaná funkce a poté program pokračuje standardně dále. Volaná funkce neosahuje žádný parametr a nevrací žádnou hodnotu. V paměti je položka reprezentovaná strukturou *MItemF*, která obsahuje navíc ukazatel na volanou funkci.

Funkce použitá pro vytvoření položky:

```
MItemF* addItemF(char *title, void(*fce)(void));
```

`*title` - Název nové položky

`*fce` - Ukazatel na funkci, se kterou položka pracuje

Definice položky – výčtový typ

Položka pracuje s proměnnou typu *integer*. Má předdefinovaný výčet hodnot, kterých může nabývat. Každá hodnota je v menu reprezentována textem. Při aktivacích položky se postupně přepíná mezi definovanými hodnotami. Položka je uložená ve struktuře *MItemE*, která obsahuje navíc definované hodnoty a jejich textové reprezentace v menu.

Funkce s proměnným počtem parametrů pro vytvoření položky:

```
MItemE* addItemE(char *title, int* val, int nrItem, ...);
```

`*title` - Název položky menu

`nrSub` - Počet definovaných hodnot

`...` - Výčet jednotlivých hodnot ve formátu: *„text“, hodnota*

4.3.4 Rozšíření o nový typ položky

Jako příklad pro rozšíření menu implementuji položku pracující s proměnnou typu *float*.

Položka bude uložena ve struktuře *MItemV*. Pro zobrazení a práci s položkou v menu je potřeba vytvořit obslužné funkce:

```
char* fltToString(MItemV* Item);
```

- Funkce vrací hodnotu proměnné položky menu jako *string*.

```
boolean fltCheckInputStr(MItemV* Item, char *in, boolean set);
```

- Funkce vrací, zda vstupní řetězec *in*, lze převést na hodnotu typu *float*. Parametr *set* určuje, zda má být aktuální hodnota také změněna.

```
boolean fltVarCompare(void* val1, void* val2);
```

- Funkce vrací, jsou-li hodnoty *val1* a *val2* rovny.

Posledním krokem je funkce, která vytvoří novou strukturu *MItemV*, naplní její proměnné podle vstupních parametrů a přiřadí ukazatele na všechny obslužné funkce:

```
 MItemV* addItemFl(char *title, float *pi, int lenght, float min, float max,
boolean editable){
            MItemV* Item = (MItemV*)malloc(sizeof(MItemV));
            float *vMin = (float*)malloc(sizeof(float));
            float *vMax = (float*)malloc(sizeof(float));
            *vMin = min;
            *vMax = max;
            Item->title = title;
            Item->pVar = pi;
            Item->pVarCp = (float*)malloc(sizeof(float));
            Item->getItemLen = varGetItemLen;
            Item->getMenuStr = varGetMenuStr;
            Item->itemActivate = varItemActivate;
            Item->itemRefresh = varItemRefresh;
            Item->editable = editable;
            Item->pMax = (void*) vMax;
            Item->pMin = (void*) vMin;
            Item->lenght = lenght;
            Item->toString = fltToString;
            Item->checkInputStr = fltCheckInputStr;
            Item->out = (char*)malloc((sizeof(char)*(Item->lenght + 1)));
            Item->varCompare = fltVarCompare;
            return Item;
      }
```

Funkce *addItemFl* se použije pro konfiguraci složení nového okna menu.

4.3.5 Obslužné funkce menu

Implementace některých funkcí, především pro sériovou komunikaci, se u různých typů mikropočítačů liší. Takové funkce, které jsou potřeba pro chod textového menu, jsem nadefinoval a jejich implementaci je třeba v použité aplikaci provést.

Jedná se o těchto pět funkcí:

`void serialInit(long bautRate)`	- inicializace sériové linky
`boolean serialAvaible()`	- jsou připravena data pro čtení
`char serialRead()`	- přečtení znaku
`void serialWrite(char* znak)`	- odeslání řetězce znaků
`void delay(int ms)`	- zpoždění v ms

4.3.6 Knihovna funkcí pro vytvoření menu

Všechny deklarace datových typů a funkcí potřebných pro chod textového menu jsou uloženy v knihovně: *menu_lib.h* a jejich definice v souboru *menu_lib.c*. Při implementaci textového menu se vytvořená knihovna připojí ke stávající aplikaci.

```
#include "menu_lib.h"
```

V hlavičkovém souboru jsou definované konstanty, které ovlivňují chod a zobrazení menu.

Maximální hloubka zanoření u vytvořeného menu.

```
#define deepMenu 5
```

Definice znaků pro vykreslení rámečku u okna menu.

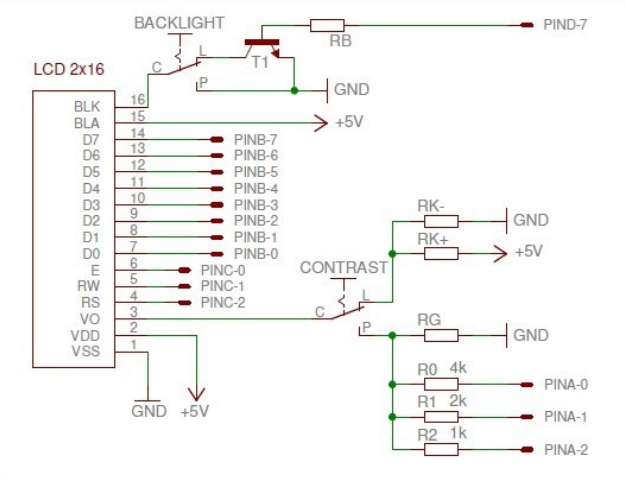
```
#define lu_c       218   //levy horni roh '+'
#define ld_c       192   //levy dolni roh
#define ru_c       191   //pravy horni roh
#define rd_c       217   //pravy dolni roh
#define h_l        196   // horizontalni linka '-'
#define v_l        179   // vertikalni linka '|'
```

Definice kláves používaných při ovládáni menu

```
#define  K_UP    'A'     //pohyb nahoru
#define  K_DOWN  'B'     //pohyb dolu
#define  K_ENTER 13      //konec editace
#define  K_ACT   ' '     //aktivace polozky
#define  K_EXIT  3       //ukonceni menu
#define  K_ESC   27      //navrat
#define  K_BSP   8       //backspace
```

5. Realizace prototypové aplikace

K realizaci prototypové aplikace jsem použil školní výukový AVR-KIT s modulem textového LCD displeje. Řídicí jednotkou AVR-KITU je mikropočítač ATmega32, který disponuje 32 kB paměti pro program a 2kB operační paměti SRAM [6]. K dispozici má také UART rozhraní, které je možné použít pro komunikaci s terminálem. Pro vývoj jsem použil volně dostupné Atmel studio s překládačem AVR-gcc, určené pro jazyk *C*.



Obr. 6: Zapojení LCD modulu [7]

Pro účely testování byla vytvořená jednoduchá aplikace, kde se text pohybuje po prvním řádku displeje. Úkolem pro textové rozhraní, bude měnit rychlost pohybu textu, zapínat a vypínat podsvícení displeje a nastavovat kontrast pomocí DA převodníku, realizovanému pomocí rezistorů připojených na výstupy mikropočítače, viz Obr. 6. Jako další možnost je volání funkce, která nastaví parametry zobrazení do výchozího stavu.

V aplikaci programu byly implementované proměnné a funkce, které ovlivňují zobrazení LCD displeje. Úkolem vytvořeného menu bude pracovat s těmito proměnnými.

```
int rychlost;          - rychlost pohybu textu po displeji v rozmezí 1 – 100.
int kontrast;          - kontrast displeje v rozmezí 0 – 7.
int podsviceni;        - podsvícení displeje 0 / 1.
void setDefault();     - nastavení výchozích hodnot pro zobrazení.
float desetine;        - desetinné číslo
```

5.1 Připojení knihovny a definice obslužných funkcí

Prvním krokem je připojeni knihovny funkcí pro vytvoření menu a nadefinovaní obslužných funkcí pro komunikaci.

- Připojení knihovny ke stávající aplikaci

```
#include "menu_lib.h"
```

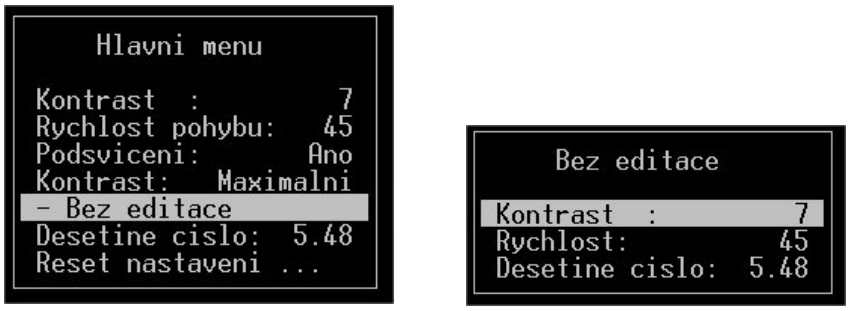
- Implementace obslužných funkcí

```
  /** inicializace serioveho rozhrani */
void serialInit(long bautRate){
      uart_init();
}

  /** odeslání pole znaků */
void serialWrite(char* znak){
      for(int i=0; znak[i] != 0; i++){
            while(!(UCSRA & (1<<UDRE)) ){}
            UDR = znak[i];
      }
}

  /** data pro čtení ze serioveho rozhrani */
boolean serialAvaible(){
      if(!(UCSRA & (1<<RXC))) return 0;
      return 1;
}

  /** čtení jednoho znaku */
char serialRead(){
      return UDR;
}

  /** zpoždění v ms */
void delay(int ms){
      delay_ms(ms);
}
```

5.2 Vytvoření menu

Pro vytvoření menu se nejdříve deklaruje ukazatel na strukturu *Menu*. Vytvoření menu provedeme pomocí funkce *CreateMenu(MItemM* screen)*. V parametru pomocí vytvořených funkci definujeme složení celého menu.

```
Menu *menu;
menu = CreateMenu(
          addSubM("Hlavni menu ",7,
              addItemI("Kontrast  :", &kontrast,1,0,7, true),
              addItemI("Rychlost pohybu:", &rychlost, 4, 1, 100, true),
              addItemE("Podsviceni:", &podsviceni, 2, "Ano", 1,"Ne", 0),
              addItemE("Kontrast:",&kontrast, 4,"Maly", 1,"Stredni", 3,
              "Velky", 5,"Maximalni", 7),
              addSubM("Bez editace ", 3,
                    addItemI("Kontrast  :", &kontrast, 1, 0, 7, false),
                    addItemI("Rychlost:", &rychlost, 4, 1, 100, false),
                    addItemFl("Desetine cislo:", &desetine, 5,0,0, false)
                    ),
              addItemFl("Desetine cislo:", &desetine,5,-15.2,85.2, true),
              addItemF("Reset nastaveni ...", setDefault)
              )
       );
```



Obr. 7: Vytvořené menu

5.3 Zajištění funkčnosti menu

Pro funkčnost menu jet třeba zajistit cyklické opakování funkce *RunMenu(*Menu)*. Pro tento účel jsem použil přerušení generované od časovače *Timer0*.

- Vektor pro obsluhu přerušení časovače 0 s funkcí zajišťující chod menu

```
 /** Vektor přerušení při přetečení čítače/časovače "0" */
ISR(TIMER0_OVF_vect)
{
      MenuRun(menu);
}
```

- Nastavení a povolení příslušného přerušení v těle programu

```
TIMSK = 0b00000001; //vybereme přerušení od TC0
TCCR0 = 4;          //spusť časovač "0" s předděličkou 256
sei();              //povol přerušení
```

5.4 Test prototypové aplikace

Výsledný program byl přeložen a nahrán do AVR-Kitu s připojeným modulem LCD displeje. Parametry připojení použitého emulátoru terminálu:

- Rozhraní: COM4
- Rychlost: 57600 baud
- Datové bity: 8
- Parita: žádná
- Stop-bity: 1
- Řízení toku: žádné

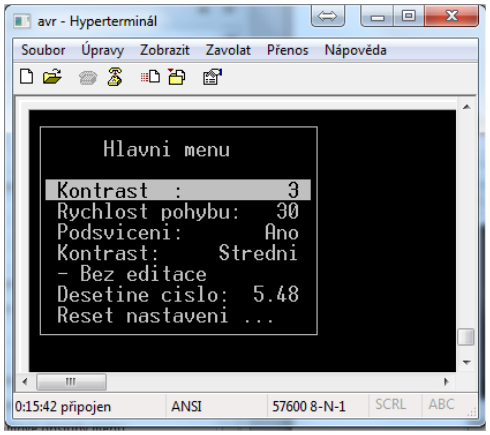
Nastavené parametry pro komunikaci musí korespondovat s parametry v mikropočítači.

Po spuštění programu je rozhraní neaktivní. Jeho aktivace se provede klávesami *CTRL+C*. Po aktivaci se zobrazí úvodní obrazovka menu. Menu lze pomocí šipek procházet. Editovat položky lze podle nastavení jen v hlavní obrazovce. Provoz menu lze ukončit opětovně *CTRL+C*.

Velikost jednotlivých obrazovek menu se dynamický mění podle délky jednotlivých položek. Při zadávání nové hodnoty parametru jsou aktivní pouze číselné klávesy. Po potvrzení nové hodnoty parametru, se kontroluje, zda je ve vymezeném rozsahu.

5.4.1 Test aplikace v terminálu Hyperterminál

V aplikaci nastavené parametry portu a emulace ANSI terminálu. Po aktivaci *Ctrl+C* se menu vykreslilo správně. Listování pomocí šipek i zanoření do pod-menu bylo v pořádku. Při editaci proměnných je kontrolována délka i formát vstupního řetězce. Funkce pro reset nastavení proběhla také v pořádku. Při změně hodnoty proměnné se menu průběžně aktualizuje.



Obr. 8: Test aplikace – Hyperterminál

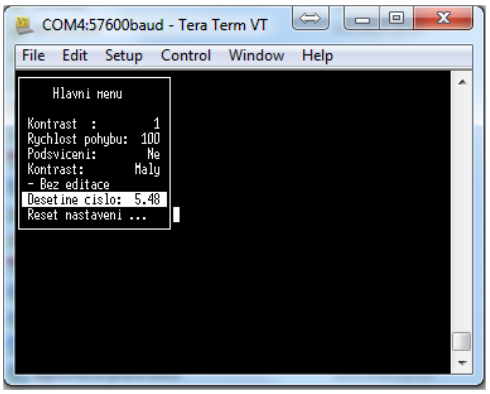
5.4.2 Test aplikace v terminálu Putty

Nastaven sériový port a parametry komunikace. Znaková sada UTF-8 změněna na CP345, původní nastavení nezobrazuje korektně znaky rámečku. Zobrazení i všechny funkcionality menu byly v pořádku.

Obr. 9: Test aplikace - Putty

5.4.3 Test aplikace v terminálu TeraTerm

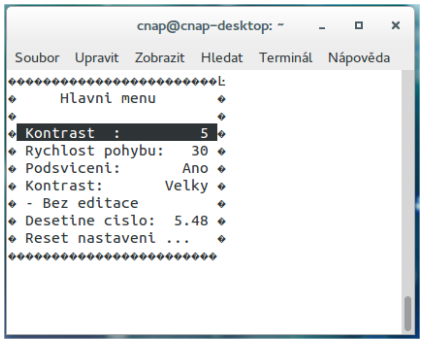
Nastaven sériový port a parametry komunikace. Ostatní nastavení ponechány ve výchozím stavu. Zobrazení i všechny funkcionality menu byly v pořádku.



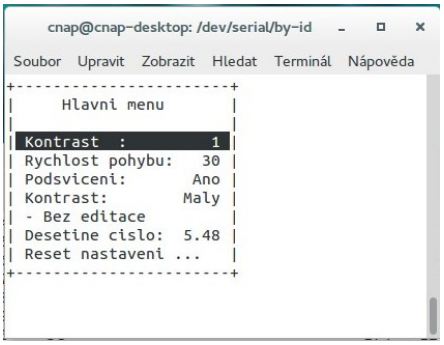
Obr. 10: Test aplikace – TeraTerm

5.4.4 Test aplikace v terminálu Minicom

Použitá aplikace Minicom v operačním systému Linux Ubuntu. Terminál využívá kódování UTF-8 a grafické znaky rámečku nezobrazoval korektně, viz Obr. 11. V aplikaci jsem je proto vyměnil za znaky z první poloviny ASCI tabulky. Zobrazení i všechny funkcionality menu byly v pořádku.



Obr. 11: Test aplikace – Minicom, chybné zobrazení



Obr. 12: Test aplikace – Minicom, správné zobrazení

5.4.5 Paměťová a procesorová náročnost

- Paměť programu – V paměti programu jsou uložené funkce pro vytvoření a obsluhu menu. Testovaná aplikace zabírala 13 kB paměti programu.

- Operační paměť – Operační paměť obsahuje především stavová data aktuální obrazovky menu. Testovaná aplikace si při kompilaci vyhradila 200B RAM paměti.

- Výkon procesoru – Zatížení procesoru ovlivňuje délka intervalu, v kterém se kontroluje výstup z klávesnice a následná komunikace po sériovém rozhraní.

5.4.6 Složitost implementace

Implementace vytvořeného textového rozhraní není složitá. Rozhraní bylo navrženo tak, aby bylo možné jednoduše zakomponovat do stávající aplikace běžící na mikropočítači. Celá implementace se skládá ze čtyř částí:

- připojit knihovnu se zdrojovým kódem rozhraní.
- definovat funkce potřebné pro sériovou komunikaci.
- pomocí funkcí nadefinovat podobu menu a inicializovat připravenou strukturu.
- zajistit cyklické volání funkce pro obsluhu menu.

Pro implementaci menu je potřeba, aby mikropočítač i s aplikací disponoval volnými hardwarovými prostředky. Jedná se o 13 kB paměti programu, 200 B RAM paměti a sériové rozhraní ke komunikaci s terminálem. Cyklické vykonávání funkce pro obsluhu menu lze zajistit přímo v aplikaci, nebo obsluhou přerušení některého z časovačů mikropočítače.

6. Závěr

Cílem mé bakalářské práce bylo porovnat možnosti jednotlivých textových rozhraní, seznámit se vlastnostmi textových terminálů a navrhnout vlastní textové rozhraní vhodné pro mikropočítače.

Pro porovnání jsem vybral textová rozhraní od nejjednodušší formy příkazového řádku až po okenní menu s pokročilými ovládacími prvky. U každého z typu jsem srovnával jeho hardwarové nároky, nároky kladené na uživatele a příklady vhodného použití. Typ použitého rozhraní je dán především aplikaci, v níž je nasazeno.

Ke komunikaci s textovým rozhraním mikropočítače jsem vybral několik volně dostupných emulátorů terminálu. Jejich společným prvkem byla podpora normy VT100/ANSI, která definuje seznam příkazů, tzv. escape sekvencí, určených pro formátování výstupu na obrazovce terminálu.

Vytvořené textové rozhraní jsem navrhl ve formě nabídkového menu, které ve svých položkách pracuje s proměnnými v běžící aplikaci. Editací těchto proměnných lze následně běh aplikace ovlivňovat. Implementace navrženého rozhraní zahrnuje připojení vytvořené knihovny, konfiguraci menu a zajištění cyklického volání funkce, která obsluhuje komunikaci s terminálem.

Obsluha menu je navržená tak, aby se minimalizoval dopad na běžící aplikaci. Ve vytvořené knihovně jsou definovány konstanty, které určují funkční klávesy pro ovládání a grafickou podobu rámečku okna. Velikost okna se dynamicky mění podle zobrazovaných hodnot. Knihovna obsahuje také funkce pro snadnou a intuitivní konfiguraci menu.

Pro realizaci jsem použil programovací jazyk *C*, který podporuje většina vývojových prostředí pro mikropočítače. Zkušební aplikaci jsem vytvořil pomocí AVR-Kitu. V testovaných terminálech menu pracovalo korektně. Problém způsobovalo zobrazení grafických znaků. Bylo potřeba v terminálu zvolit znakovou sadu obsahující příslušné symboly, nebo změnit grafické znaky v aplikaci mikropočítače.

V bakalářské práci jsem splnil všechny vytyčené cíle. Navržené rozhraní je kompletní a funkční a může být snadno rozšířeno o další datové typy. Jako další typy položek bych navrhl například datum, čas nebo text. Zobrazení okna menu by mohlo být rozšířeno o nápovědu nebo krajní meze při editaci proměnné.

Literatura

1. **Electronics, MCS.** *BASCOM-AVR.* [Online] 2014. [Citace: 7. květen 2015.] http://avrhelp.mcselec.com/index.html?asc.htm.

2. **Lawyer, David S.** Text-Terminal-HOWTO. [Online] Leden 2010. [Citace: 7. květen 2015.] http://www.linuxdoc.org/HOWTO/Text-Terminal-HOWTO.html.

3. **Wikipedia.** ANSI escape code. [Online] 12. březen 2015. [Citace: 7. květen 2015.] http://en.wikipedia.org/wiki/ANSI_escape_code.

4. **TechNet, Microsoft.** HyperTerminal. [Online] 21. leden 2005. [Citace: 7. květen 2015.] https://technet.microsoft.com/en-us/library/cc784492(v=ws.10).aspx.

5. **Penalver, Christopher M.** Minicom. *Ubuntu documentation.* [Online] 20. leden 2015. [Citace: 7. květen 2015.] https://help.ubuntu.com/community/Minicom.

6. **Corporation, Atmel.** ATmega32 Summary. [Online] únor 2011. [Citace: 7. květen 2015.] http://www.atmel.com/images/2503s.pdf.

7. **Olivka Petr, Seidl David.** *Návody do cvičení pro předmět: Architektury počítačů a paraelních systémů.* Ostrava : VŠB-TU FEI, 2014.

8. **Pechal, Stanislav.** *Monolitické mikropočítače.* Praha : BEN - technická literatura, 1990. ISBN;80-86056-30-9.

Seznam obrázků

Přílohy

Elektronická příloha - CD se zdrojovými kódy a aplikaci:

`\knihovna\menu_lib.h` `\memu_lib.c`	-	Knihovna pro implementaci menu
`\projekt\`	-	Složka s demo projektem pro Atmel Studio
`\bakalarska_prace.pdf`	-	text bakalářské práce